DETAILED ACTION

A. This action is in response to the following communications: Amendment filed: 03/30/2009. This action is made **Final**.

B. Claims 1-2, 6, 8-13 and 15-32 remain pending.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

2. Claims 1-2, 6, 8-11 are rejected under 35 U.S.C. 102(e) as being anticipated by Cull, Christopher A. et al. (US Pub. 2003/0214533 A1), herein referred to as "Cull".

As for claims 1-2, 6, 8-11 the disclosure of Cull will show a system which features two different graphical technologies (graphic systems) being merged on a client computer where these graphic system correspond to three different type of graphic windows. On the client there are two type of windows being rendered for the user of the computer wherein one window is a dummy (mock) window which does not access anything from the operating system; instead it retrieve a bitmap from Cull's system

which features a module called VMM (visual merge management) to display to the user, any controls, any interaction, etc... is ignored by this window as it functions as a dummy window that is using no device context. Below (overlapped) this window (dummy window) is a control window (92) which hooks commands from the user input device where this type of window uses device context. These two windows are used to show a "viewport" into a complex system which is not being ran on clients computer but instead on a more advanced computing machine (80; 3D visual system). The result of this disclosure provides the same functionality as described by the current claim language as found in at least the cited portions of Cull.

As claim 1, Cull teaches a system, embedded at least in part on tangible computer readable medium for enabling interoperability between two graphics technologies (par.28), comprising: a first graphics system configured to render window content in a first mode (par.29) the first graphics system being further configured to reference a first type of window using a window handle associated with an instance of the first type of window (figure 3; par.29-30); a second graphics system configured to render windows in a second mode (figure 3-4; par.31-33), the second graphics system being further configured to reference a second type of window without using any window (par.34); and an interoperability component configured to cause a dummy window handle to be created for an instance of a window of the second type, wherein a null device context is associated with the dummy window handle to facilitate a lookup of the second type of window, wherein any drawing done to null device context is lost (par.35-36); Cull provides essentially a dummy window which does not have a

purpose other than being visually displayed, it does not accept input, it does not access data and does not use device context; further Cull teaches a window that is positioned underneath of this dummy window which is used for interoperability to the complex 3D visuals system (80) as the dummy window cannot provide any controls, data, etc… to the 3D visuals system (figure 4); thus Cull provides clear support of a mechanism and technique for providing interoperability between two different graphics technologies.

As claim 2, Cull further teaches an application program including a first

window and a second window (figure 3), the first window being of the first type and the second window being of the second type (figure 3; par.29-32).

As claim 6, Cull further teaches the second graphics system is configured to

create a mapping from the dummy window handle to a node in an internal construct used by the second graphics system to manage windows of the second type (par.34-36).

As claim 8, Cull further teaches the second graphics system is further

configured to create a render target for receiving rendered window content (figure 4; par.34-35).

As claim 9, Cull further teaches the render target resides in system memory

(figure 4).

As claim 10, Cull further teaches the render target resides in video memory

(figure 4; par.34).

As claim 11, Cull further teaches the render target records rendering

commands generated for windows of the second type and that are played back during composition to generate display output (figure 3-4; par.29-35).

3. Claims 12-13 and 15-32 are rejected under 35 U.S.C. 102(e) as being anticipated by Gershony et al. (US 6,549,218), herein referred to as "Gershony".

As claim 12, Gershony teaches a tangible computer-readable storage medium (fig. 1, label 32) having computer executable components (col. 4, lines 65-67; col.5, lines 1-2) for enabling interoperability between two graphics technologies (col. 2, lines 44-55), comprising: a first graphics system that comprises an immediate mode graphics technology; a second graphics system hat comprise a compositional mode graphics

technology (figure 3, col.7, lines 33-59) an interoperability component that interfaces with an application program (col. 2, line 67; col. 3, lines 1-4), the application program including a first window and a second window (col. 1, lines 34-37), the first window being compatible with the first graphics system that uses window handles to reference windows (fig. 3, labels 340 and 350; col. 7, lines 60-64: col. 8, lines 13-15), the second window being compatible with a second graphics system that does not use window handles (fig. 3, label 340; col. 7, lines 60-64, that if the window is redirected it will not utilize the same window handle as depicted for the first window, to ensure the window is redirected); and a mock window handle associated with the second window, the mock window handle to indicate that the second window is compatible with the second graphics system, wherein a null device context is associated with the dummy window handle to facilitate a lookup of the second type of window, wherein any drawing done to null device context is lost (fig. 3, label 320; col. 6, lines 61-65; col. 7, lines 33-41; col. 6, lines 14-15; col. 8, lines 52-58, that using "MICROSOFT WINDOWS" to create window "CreateWindowEX 0", using known "Microsoft Component Object Model (COM) to call functions "Microsoft Windows GetDCo()" with a NULL window handle as a parameter and "CreateCompatibleBitmap0" to create a dummy (blank/mock) Window bitmap in memory, which is compatible with (e.g., has the same color depth as) the screen device context and to call "ViewObject2::Draw () to draw (perform) a graphics related action to enhance). Null device context (a place holder to look up the graphic visual of an element (e.g. window). (col.6, lines 61-67; col.7, lines 1-13); When, how and/or where drawing is lost is not explained in the specification, thus when the style bit is written to a

different value the value before the new value is lost, in such the style bit will be lost and drawing to it is lost).

As claim 13, Gershony further teaches a mapping, maintained by the second graphics system, from the mock window handle to a node in an internal construct used by the second graphics system to manage the second window (col. 9, lines 45-5 I, that a data structure will contain mapping linking the mock window handle to the node and is a key module to managing the display of windows).

As claim 15, Gershony further teaches the second graphics system is further configured to create a render target for receiving rendered window content (col. 6, lines 61-67; col. 7, lines 1-13).

As claim 16, Gershony further teaches the render target comprises a software • render target (fig. 1, label 22; col. 6, lines 61-67; col. 7, line 1).

As claim 17, Gershony further teaches the render target comprises a hardware render target (fig. 1, labels 22, 47 and 48; col. 6, lines 61-67; col. 7, line, that there must

be video memory as described as known better and image (target) can be sent to the display monitor, it must be buffered to an area of video memory).

As claim 18, Gershony further teaches the render target records rendering commands generated for the second window and that are played back during composition to generate display output (col. 6, lines 61-67; col. 7, lines 1-13; col. 8, lines 13-29, that by applying the special effects to the window the final result will be displayed (played back)).

As claim 19, Gershony further teaches the mock window handle is associated with a device context associated with the second window (col. 2, lines 11-16).

As claim 20, Gershony further teaches the device context comprises a null device context (col. 8, lines 51-53, lines 66-67; col. 9, lines 1-8, that if the function fails, the return value is null, indicating an error or an invalid HWND parameter).

As claim 21 (Currently Amended), Gershony teaches a computer-implemented method (fig. 1, labels 36, 37) for enabling interoperability between two graphics technologies (col. 2, lines 44-55), comprising: receiving a request to create a new

window (col. 2; lines 11-16); determining if the new window is of a type associated with an alternative graphics system that does not require the use of a window handle (fig. 3, label 340; col. 7, lines 62-64); if the new window is of a type associate with an alternative graphics system, creating a dummy window handle for the new window to facilitate interoperability with a conventional graphics system, wherein a null device context is associated with the dummy window handle to facilitate a lookup of the second type of window, wherein any drawing done to null device context is lost (fig. 3, label 320; col. 6, lines 61-65; col. 7, lines 33-41; col. 6, lines 14, 15; col. 8, lines 52-58, that using "MICROSOFT WINDOWS" to create window "CreateWindowEX ()", after that using known "Microsoft Component Object Model (COM) to call functions "Microsoft Windows GetDCo()" with a NULL window handle as a parameter to create a blank (dummy) window bitmap in memory); creating a new visual to be created in connection with the new window, the visual being a construct associated with the alternative graphics system (col. 8, lines 26-34; calling function "CreateCompatibleBitmap()" to make a dummy (blank) Window bitmap in memory, which is compatible with (e.g., has the same color depth as) the screen device context and calling "ViewObject2::Draw () to draw (perform) a graphics related action to enhance); and associating the dummy window handle with the new visual (fig. 3, label 340; col. 7, lines 60-64, that if the window is redirected it will not utilize the same window handle as depicted for the first window, to ensure the window is redirected). Null device context (a place holder to look up the graphic visual of an element (e.g. window). (col.6, lines 61-67; col.7, lines 1-13); When, how and/or where drawing is lost is not explained in the specification, thus when the

style bit is written to a different value the value before the new value is lost, in such the style bit will be lost and drawing to it is lost).

As claim 22 (Currently Amended), Gershony further teaches if the new window is not of the type associated with the alternative graphics system, rendering the window in accordance with a conventional graphics system (fig. 3, labels 350, 360, 370; col. 8, lines 13-19).

As claim 23 (Currently Amended), Gershony further teaches receiving an instruction to render display content to the new window referenced by the dummy window handle (col. 3, lines 8-12), looking up the new visual based on the association between the dummy window handle and the new visual (col. 8, lines 43-45, that applying visual effects, is only accomplished by reading/referencing the window handles), and rendering the display content to the new visual (fig. 3, labels 350, 360, 370).

As claim 24, Gershony further teaches rendering the display content to the new visual (fig. 3, labels 350, 360, 370) further comprises issuing rendering commands to a render target associated with the new visual (col. 3, lines 8-12).

As claim 25, Gershony further teaches the render target comprises a software render target (fig. 1, label 22; col. 6, lines 61-67; col. 7, line 1).

As claim 26, Gershony further teaches the render target comprises a hardware render target (fig. 1, labels 22, 47 and 48; col. 6, lines 61-67; col. 7, line, that there must be video memory as described as known before and image (target) can be sent to the display monitor, it must be buffered to an area of video memory).

As claim 27, Gershony further teaches the render target records rendering commands generated for the new window that are played back during composition to generate display output (col. 6, lines 61-67; col. 7, lines 1-13; col. 8, lines 13-29, that by applying the special effects to the window the final result will be displayed (played back)).

As claim 28, Gershony further teaches a computer-readable medium encoded with computer-executable instructions for performing the method of claim 21 (fig. 1, label 36; col. 5, lines 3-7).

As claim 29, Gershony further teaches the system recited in claim 2, wherein the first mode comprises a compositional mode of graphics technology (figure 3, col.7, lines 33-59).

As claim 30, Gershony further teaches the system recited in claim 2, wherein the second mode comprises an immediate mode of graphics technology (figure 3, col.7, lines 33-59).

As claim 31, Gershony further teaches the system recited in claim 6, wherein the internal construct comprises a visual tree, and the node comprises a visual (col.8, lines 34-41; child and parent relationship make up a hierarchy of windows (z-order; layering) col.1, lines 45-51).

As claim 32, Gershony further teaches the computer-readable medium recited in claim 13, wherein the internal construct comprises a visual tree, and the node comprises a visual (z-order; layering) col.1, lines 45-51).

---

4. Claims 1-2, 6, 8-11 are rejected under 35 U.S.C. 102(e) as being anticipated by Lupu, Corneliu I. et al. (US Pub. 2004/0100480 A1), herein referred to as "Lupu".

As claim 1, Lupu teaches a system, embedded at least in part on tangible computer readable medium for enabling interoperability between two graphics technologies (par.6-11), comprising: a first graphics system configured to render window content in a first mode (par.27) the first graphics system being further configured to reference a first type of window using a window handle associated with an instance of the first type of window (figure 5; par.29-30); a second graphics system configured to render windows in a second mode (figure 4-5; par.31-33), the second graphics system being further configured to reference a second type of window without using any window (par.27-28, 34); and an interoperability component configured to cause a dummy window handle to be created for an instance of a window of the second type, wherein a null device context is associated with the dummy window handle to facilitate a lookup of the second type of window, wherein any drawing done to null device context is lost (par.27,35-36).

As claim 2, Lupu further teaches an application program including a first window and a second window (figure 3), the first window being of the first type and the second window being of the second type (figure 5; par.27-30).

As claim 6, Lupu further teaches the second graphics system is configured to create a mapping from the dummy window handle to a node in an internal construct

used by the second graphics system to manage windows of the second type (par.27-28; 34-36).

As claim 8, Lupu further teaches the second graphics system is further configured to create a render target for receiving rendered window content (figure 4; par.27-28; 34-35).

As claim 9, Lupu further teaches the render target resides in system memory (figure 4-5).

As claim 10, Lupu further teaches the render target resides in video memory (figure 4-5; par.27-28; 34).

As claim 11, Lupu further teaches the render target records rendering commands generated for windows of the second type and that are played back during composition to generate display output (figure 5; par.27-32).

(Note :) **It is noted that any citation to specific, pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon for all that it would have reasonably suggested to one having ordinary skill in the art. In re Heck, 699 F.2d 1331, 1332-33, 216 USPQ 1038, 1039 (Fed. Cir. 1983) (quoting In re Lemelson, 397 F.2d 1006, 1009, 158 USPQ 275, 277 (CCPA 1968)).**

Response to Arguments

Applicant's arguments with respect to claims 1-2, 6, 8-11 have been considered but are moot in view of the new ground(s) of rejection.

Applicant's arguments filed 03/30/2009 have been fully considered but they are not persuasive.

After careful review of the amended claims (given the broadest interpretation) and the remarks provided by the Applicant along with the cited reference(s) the Examiner does not agree with the Applicant for at least the reasons provided below:

A1. As for claims12-13 and 15-32; Applicant argues that Gershony does not teach a null device context is associated with the dummy window handle to facilitate a lookup of the second type of window, wherein any drawing done to the null device context is lost.

R1. Examiner does not agree. As defined in the specification a null device context is only a place holder that can be used to lookup a visual (par.27 of spec.). A device context is a structure that defines a set of graphic objects and their associated attributes, as well as the graphic modes that affect output. Gershony describes a place holder that is used as a lookup that depicts how a window is rendered to the screen (visual), thus Gershony teaches a null device context (a place holder to look up the

graphic visual of an element (e.g. window). (col.6, lines 61-67; col.7, lines 1-13). Drawing to the null device context is lost, since they null device context is set throughout the system operation, the specification does not clearly disclose when, how and/or where the drawing to the null device context is lost (the style bit is set over and over again throughout the use of the system to render different effects of graphical user interface elements, thus any drawing to it will be lost since it would be re-written over and over again when a new command is request to render an new effect for a graphical user interface element throughout use of the system; col.7, lines 18-59). Thus Gershony does in fact teach the same functionality, using different terminology, a null device context is associated with the dummy window handle to facilitate a lookup of the second type of window, wherein any drawing done to the null device context is lost.

Examiner notes that amending towards the added limitation of claim 1 would be efficient to show a difference between Gershony and the immediate application.

A2. As for claims12-13 and 15-32; Applicant argues that Gershony does not teach a system configured to reference a second type of window without using any window handle.

R2. Examiner does not agree Gershony shows in column 7, lines 60-67 and column 8, lines 1-23 the process of painting windows using a bitmap and not using that of a window handle. An example of using a handle of a layered window is used if the

discloser of Gershony in column 8 which is only mentioned as an example and is later expressed as not the only embodiment in the disclosure. Further the described handle mentioned in Gershony, as pointed out by Applicant, is not that of a traditional "window handle" as argued against by the Applicant. Thus Gershony clearly teaches a system that is configured to reference a second type of window without using any window handle.

Examiner notes that amending towards the added limitation of claim 1 would be efficient to show a difference between Gershony and the immediate application.

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Inquires

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nicholas Augustine whose telephone number is 571-270-1056 and fax is 571-270-2056. The examiner can normally be reached on Monday - Friday: 9:30am- 5:00pm Eastern.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Weilun Lo can be reached on 571-272-4847. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Steven B Theriault/
Primary Examiner, Art Unit 2179

/Nicholas Augustine/
Examiner
Art Unit 2179
2009